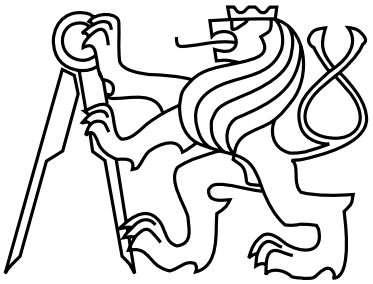
České vysoké učení technické v Praze
Fakulta stavební
Katedra geomatiky



Cvičení 1

Úloha č. 1: Geometrické vyhledávání bodu

Bc. Petr Poskočil a Bc. Marek Fáber

Vyučující: doc. Ing. Tomáš Bayer, Ph.D.

Studijní program: Geodézie a kartografie, Navazující magisterský

Obor: Geomatika

25. října 2019

Obsah

Kapitola 1

Zadání

Úloha č. 1: Geometrické vyhledávání bodu

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, ..., P_n\}$*, analyzovaný bod q.*

Výstup: P_i, $q \in P_i$.

Nad polygonovou mapou implementujete následující algoritmy pro geometrické vyhledávání:

- Ray Crossing Algorithm (varianta s posunem těžiště polygonu).
- Winding Number Algorithm.

Nalezený polygon obsahující zadaný bod q graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.

Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).

Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou reprezentaci jednotlivých polygonů použijte špagetový model.

Hodnocení:

Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně na hranici polygonu.	10b
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+2b*
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+2b*
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	*+2b*
Algoritmus pro automatické generování nekonvexních polygonů.	*+5b*
Max celkem:	**21b**

Obrázek 1.1: Zadání cičení č. 1 [1]

1.1 Údaje o bonusových úlohách

Cílem úlohy je představit grafickou aplikaci na geometrickou detekci polohy bodu vůči polygonu, která rozlišuje stavy jako uvnitř, vně, a na hraně/vrcholu. Bonusovou úlohou je myšléná širší funkcionalita aplikace. Do aplikace bylo zakomponováno oštření singulárního případu u Winding Number Algorithm - bod ležící na hraně polygonu, dále pak oštření singulárního případu u Winding Number Algorithm a Ray Crossing Algorithm - bod je totožný s vrcholem jednoho nebo více polygonů, a na nakonec grafické zvýraznění polygonů do kterých spadá bod Q včetně ošetření obou singulárních případů [1].

Kapitola 2

Popis problému

Mějme geometrický obrazec v rovině, jež se skládá z j mnohoúhelníků - polygonů P_j, polygony jsou tvořeny i vrcholy p_i a stranami σ_{P_i}. Do tohoto obrazce chceme umístit bod Q a určit jeho polohu vůči konkrétnímu polygonu, či polygonům. Při určování polohy se výpočet provádí pro každý polygon zvlášť a na základně tohoto určení se rozhodne o poloze bodu vůči polygonu ve kterém se bod Q nachází - lokální procedura. Posléze se lokální procedura opakuje pro j mnohoúhelníků - polygonů P_j na globální úrovni.

Možné scénáře polohy bodu Q na lokální úrovni:

- $Q \notin P_j$
- $Q \in P_j$
- $Q \in \sigma_{P_i}$
- $Q \equiv p_i$

Možné scénáře polohy bodu Q na globální úrovni:

- $\Sigma_{Q \in P_j} = 0; \Rightarrow Q \notin P$
- $\Sigma_{Q \in P_j} = 1; \Rightarrow Q \in P \wedge Q \notin \sigma_{P_i}$
- $\Sigma_{Q \in P_j} > 1; \Rightarrow Q \in \sigma_{P_i} \vee Q \equiv p_i$

Pro technické řešení tohoto problému je nejprve nutné úlohu převést na vztah bodu a mnohoúhelníku, při níž se opakovaně určuje poloha bodu vzhledem k mnohoúhelníku. Poté byly zvoleny dva různé výpočtení algoritmy vhodné pro úrčení polohy bodu vůči nekonvexním polygonům. Za prvé Metoda ovíjení, z anglického *Winding Algorithm*, a za druhé Paprskový algoritmus, z anglického *Ray Crossing Algorithm*. Oba tyto algoritmy disponují relativně nízkou náročností na výpočetní sílu.

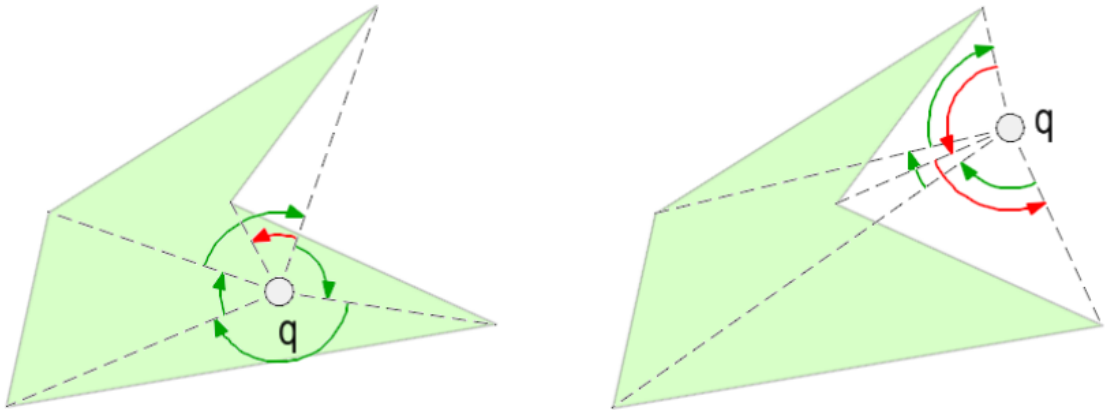
Kapitola 3

Popis použitých algoritmů

Pro určování polohy bodu Q v obrazci tvořeném polygony lze uplatnit mnoho algoritmů. Vždy je třeba zvolit vhodný algoritmus, který nebude příliš náročný na tvorbu a výpočet. Z tohoto důvodu byly zvoleny algoritmy Winding Number Algorithm a Ray Crossing Algorithm. Pro řešení úlohy by bylo možné využít také algoritmů Line Sweep Algorithm, Divide and Conquer nebo Brute Force Algorithm [2].

3.1 Winding Algorithm

Tato metoda spočívá ve sčítání úhlů, které svírají spojnice vrcholů p_i s určovaným bodem Q. Pokud se otáčíme ve směru hodinových ručiček od bodu p_i k bodu p_{i+1}, tak má úhel kladné znaménko, v opačném případě má znaménko záporné. Pokud výsledný úhel je roven 2π, tak se bod nachází uvnitř polygonu - $Q \in P_j$, jinak vně polygonu - $Q \notin P_j$.



Obrázek 3.1: Princip Winding Algorithm [2]

Při práci s algoritmy vyvstávají různé problematické situace, je tedy nuté udělat jejich rozbor a ošetřit tyto situace v kódu. K problematickým situacím při řešení této úlohy může dojít,

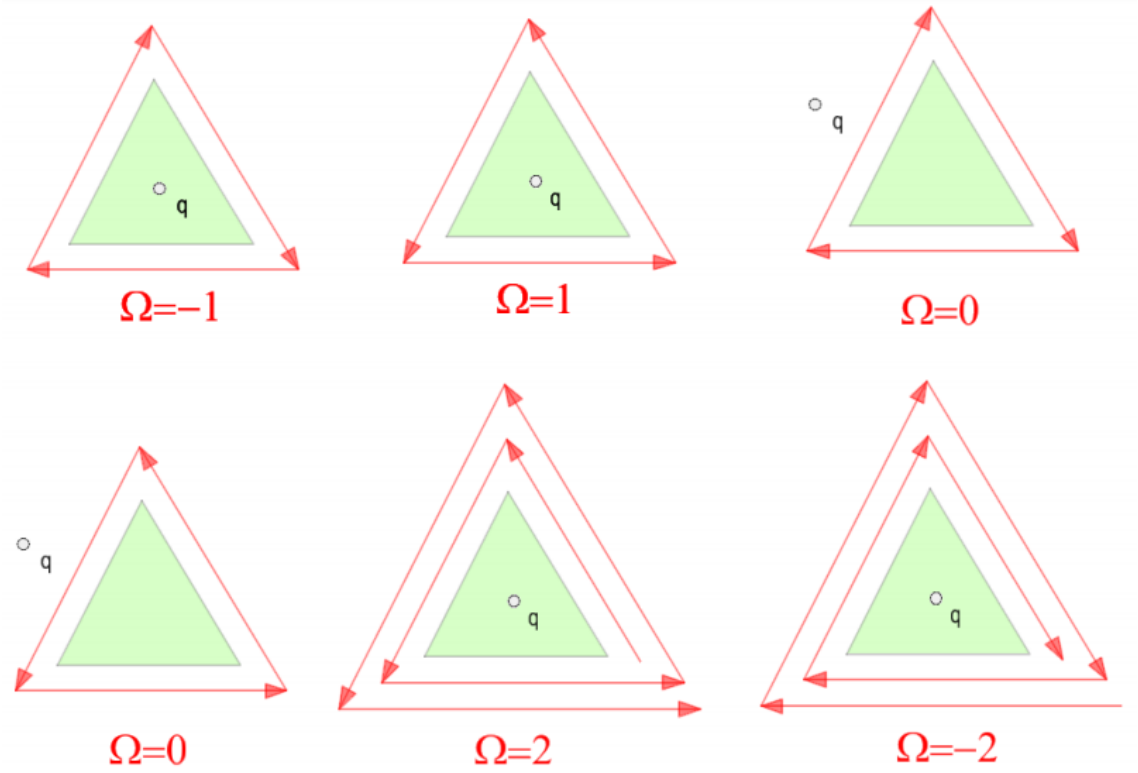
když bod leží na hraně polygonu, nebo je totožný s některým vrcholem polygonu. Takovéto situace je potřeba ošetřit, aby došlo ke správnému výběru polygonu.

3.1.1 Problematické situace u Winding Algorithm

V případě, že by bod Q ležel na některém z vrcholů polygonu, by algoritmus vyhodnotil, že bod leží mimo polygon, jelikož by součet úhlů vyšel menší než 2π. Taková situace nastane když $Q[x_Q, y_Q] \equiv p_i[x_i, y_i]$ a je potřeba v kódu ošetřit následující podmínkou: $|x_p - x_Q| < \epsilon \wedge |y_p - y_Q| < \epsilon$. Pokud je tato podmínka splněna, tak výsledkem je, že se bod nachází uvnitř polygonu - $Q \in P$.

Pro případ, když bod Q leží na hraně $|AB|$ polygonu, dojde u výpočtu, že bod leží vně polygonu, jelikož součet úhlů vyjde π nebo 3π. Tato situace se ošetřuje podmínkou, kdy platí: $(|AB| - (|AQ| + |BQ|)) < \epsilon$.

3.1.2 Implementace algoritmu



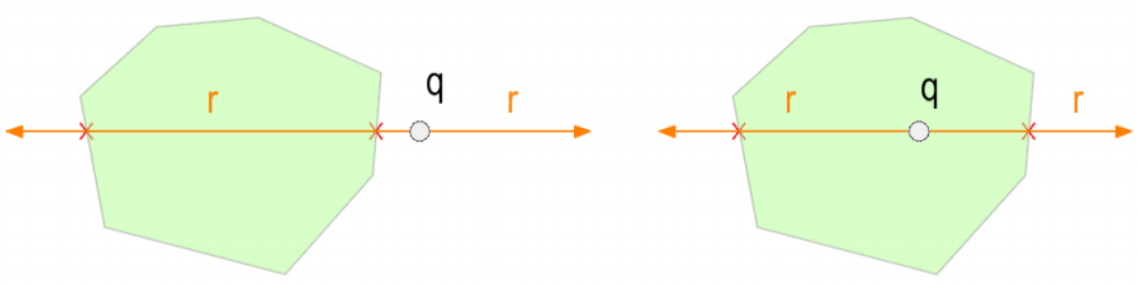
Obrázek 3.2: Princip Winding Algorithm [2]

1. Inicializace: $\omega = 0$, práh tolerance $\epsilon = 1e - 10$
2. Orientace o_i bodu Q ke straně $|p_i, p_{i+1}|$

3. Určení úhlu $\omega_i = p_i, q, p_{i+1}$
4. Pokud: $Q \in \sigma_L; \Rightarrow \omega = \omega + \omega_i$
5. Nebo: $Q \in \sigma_R; \Rightarrow \omega = \omega - \omega_i$
6. Pokud: $(\omega - 2\pi) < \epsilon; \Rightarrow Q \in P$
7. Nebo: $Q \notin P$

3.2 Ray Crossing Algorithm

Metoda Ray Crossing Algorithm vychází z určování počtu průsečíků hran polygonu a polopřímky vycházející z libovolného bodu. Když jako výchozí bod polopřímky, nebo-li paprsku, zvolíme bod Q, můžeme tak určit jeho polohu vůči polygonu. Počet průsečíků si označíme jako k. Pokud je k liché, tak bod leží uvnitř polygonu - $Q \in p[j]$, jinak leží vně - $Q \notin p[j]$.

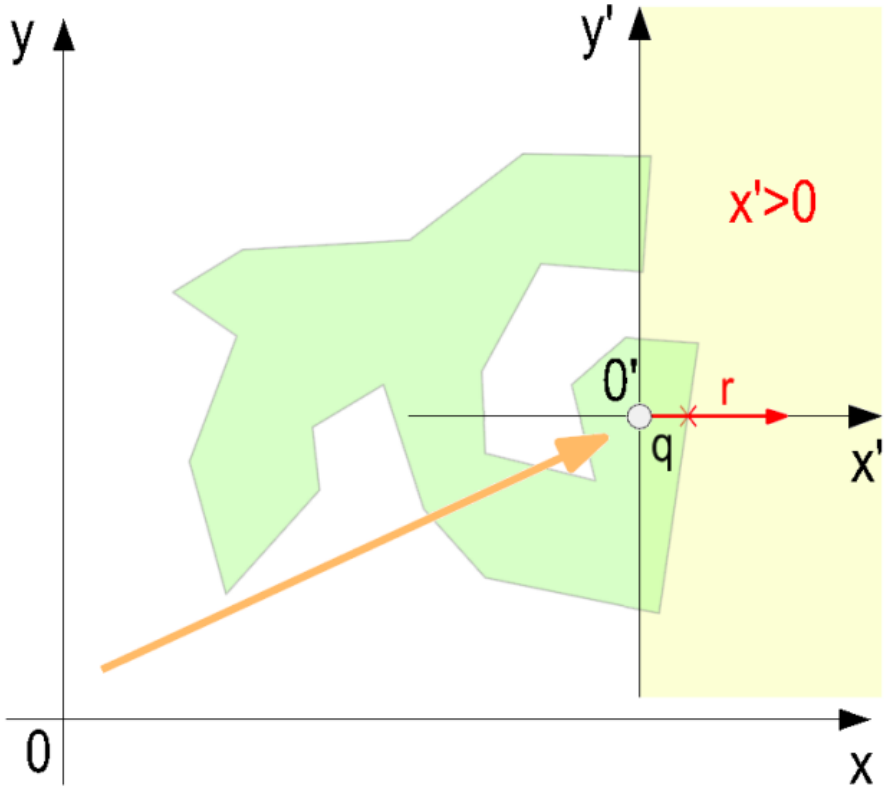


Obrázek 3.3: Princip Ray Crossing Algorithm [2]

3.2.1 Problematické situace u Ray Crossing Algorithm

Při této metodě může dojít takové k situaci, že některá hrana polygonu leží na polopřímce z bodu Q. V takovém případě dochází k tomu, že počet průsečíků polopřímky s hranami polygonu je sudý, i když je bod Q uvnitř polygonu. Z tohoto důvodu se zavádí redukovaný souřadnicový systém s počátkem v bodě Q, kde osa x' je rovnoběžná s osou x a osa y' je kolmá na x'. Řešení úlohy se poté omezí pouze na hrany, jejichž jeden vrchol leží pod osou x' a druhý nad osou x'.

U metody Ray Crossing Algorithm se problematické situace řeší principiálně stejně jako u Winding Number Algorithm, viz. 3.1.1.



Obrázek 3.4: Princip Ray Crossing Algorithm - redukované souřadnice[2]

3.2.2 Implementace metody

1. Inicializace počtu průsečíků: $k = 0$
2. Redukce souřadnic x_{p_i} vůči x_Q $x'_i = x_i - x_Q$, viz. 3.4
3. Redukce souřadnic y_{p_i} vůči y_Q $y'_i = y_i - y_Q$
4. Podmínka: $if(y'_i > 0) \wedge (y'_{i-1} <= 0) \| (y'_{i-1} > 0) \wedge (y'_i <= 0)$
5. Pokud splněna: $x'_m = (x'_i y'_{i-1} - x'_{i-1} y'_i)/(y'_i - y'_{i-1})$
6. Pokud: $x'_m > 0; \Rightarrow k = k + 1$
7. Pokud: $k\%2 = 0; \Rightarrow Q \in P$ - Sudý
8. Nebo: $Q \notin P$

Kapitola 4

Vstup a výstup aplikace

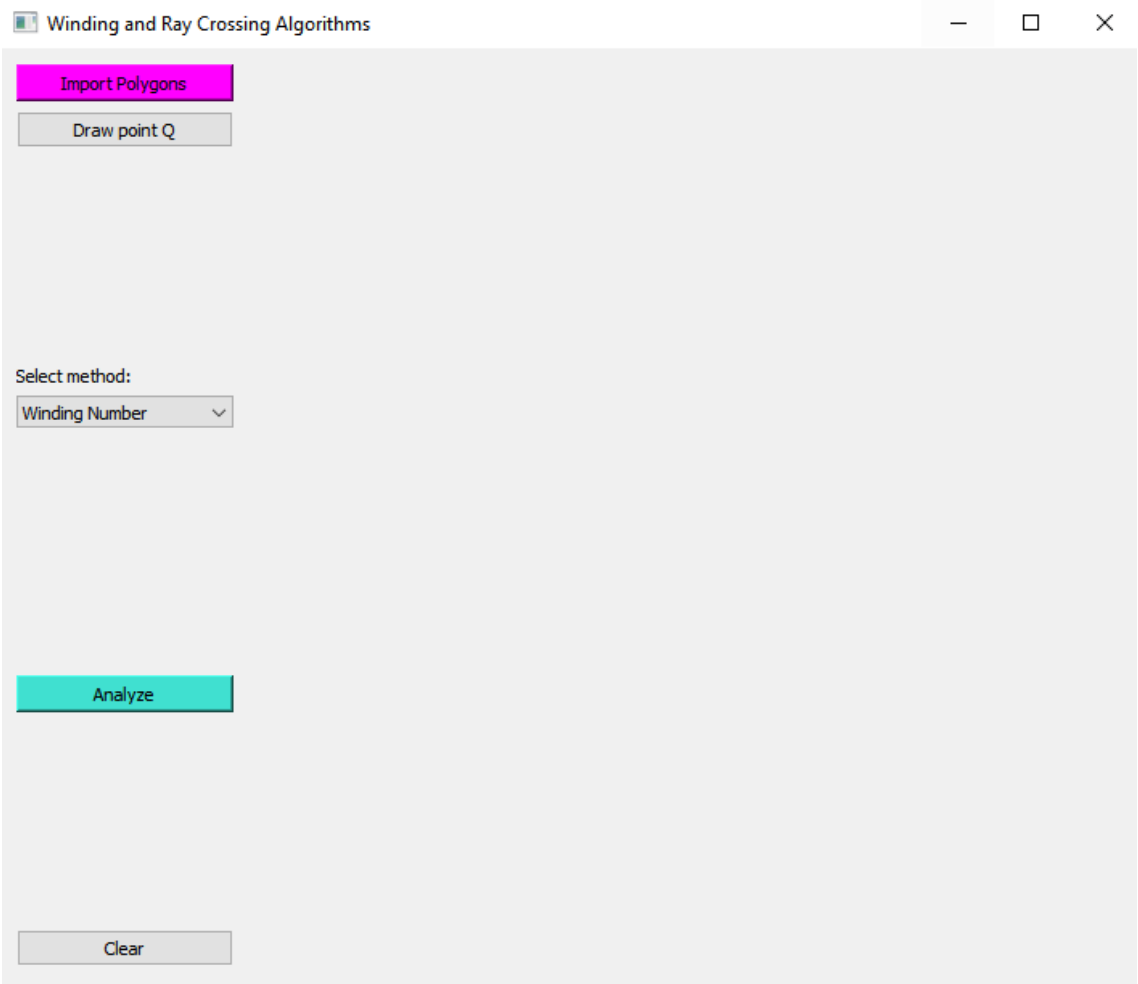
4.1 Vstup

- Vstupními daty pro tuto úlohu je textový soubor formátu .txt a bod Q. Textový soubor je potřeba do aplikace načíst pomocí tlačítka "Import Polygons". Soubor obsahuje body jednotlivých polygonů v řádcích ve formátu [číslo bodu, souřadnice X, souřadnice Y]. Každý polygon začíná bodem 1 a končí číslem n-tým bodem, pokud v souboru je načítán bod s číslem 1, dojde k ukončení předchozího polygonu a začíná nový polygon. Čísla bodů a souřadnice jsou v souboru odděleny mezerami.
- Bod Q do aplikace vloží uživatel kliknutím levým tlačítkem myši po aktivování přes tlačítko "Draw point Q".

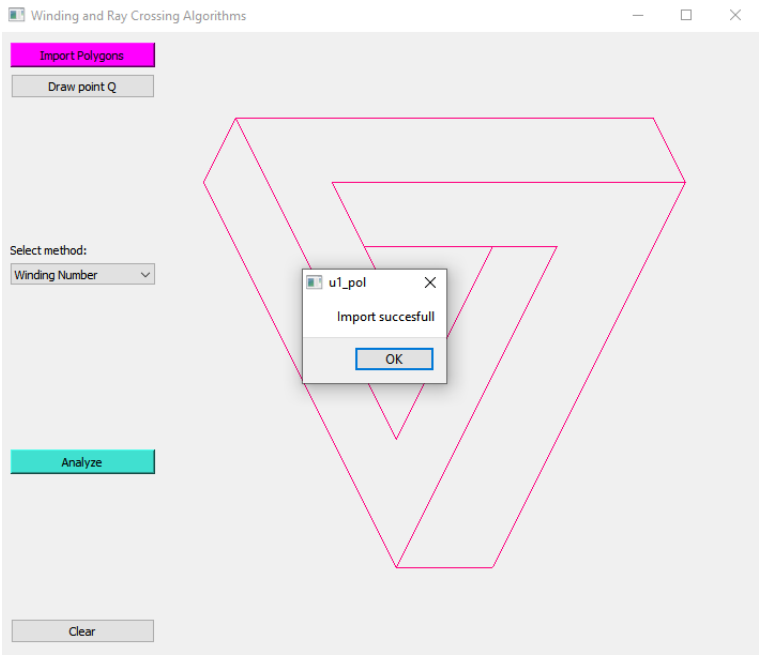
4.2 Výstup

- Výstupem úlohy je grafická aplikace, která po načtení souboru s polygony a volbě bodu Q určí polygon, ve kterém se bod nachází. Pro analýzu polohy bodů je možné zvolit ze dvou algoritmů Winding Number Algorithm a Ray Crossing Algorithm.
- Polygon, ve kterém se bod nachází, se po stisknutí tlačítka "Analyze"vybarví.
- Pokud je bod na hraně dvou polygonů nebo je totožný s vrcholem více polygonů, tak se vybarví všechny dotčené polygony.
- Pokud se bod nenachází v žádném polygonu, tak se žádný polygon nevybarví.
- Grafický výstup se pak stiskem tlačítka "Clear"smaže.

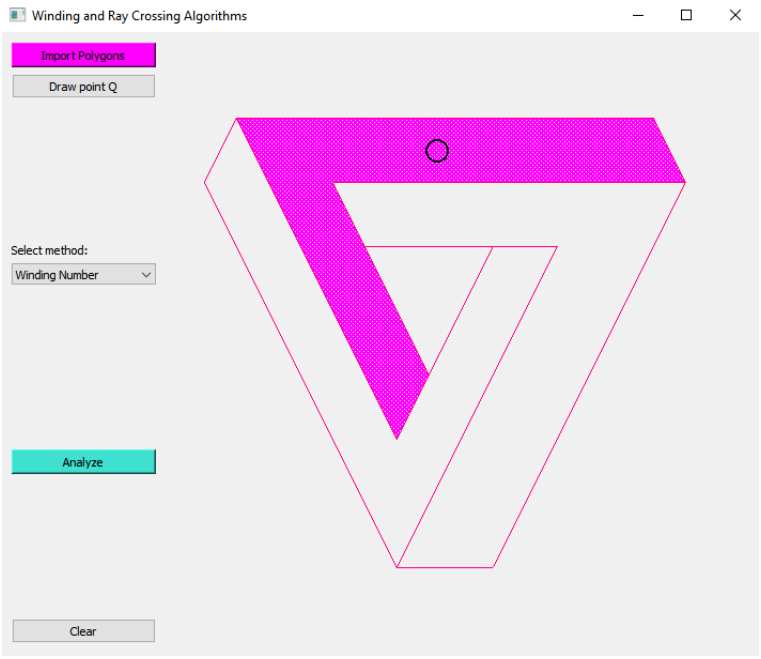
4.3 Prostředí



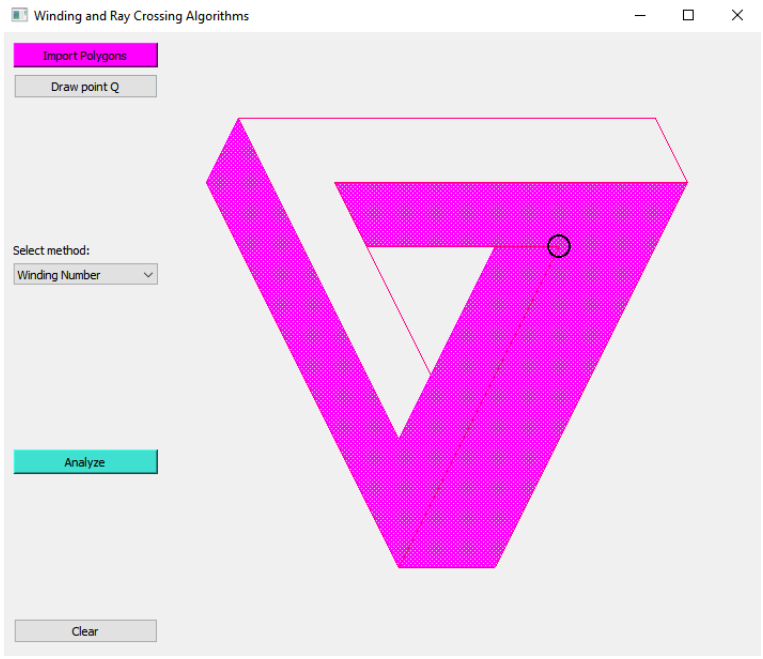
Obrázek 4.1: Prostředí po spoštění aplikcace



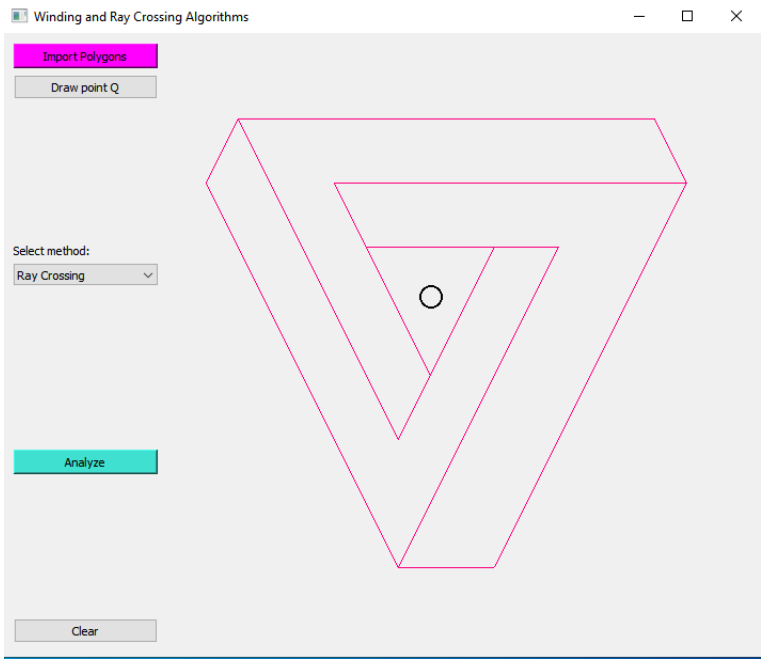
Obrázek 4.2: Hláška úspěšného importu



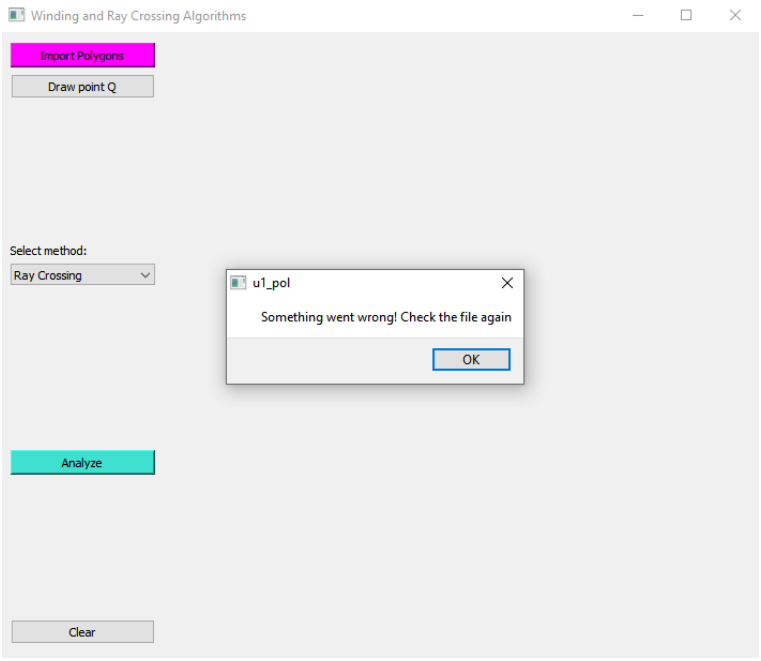
Obrázek 4.3: Vybarvení polygonu obsahujícího bod Q



Obrázek 4.4: Vybarvení dvou polygonů s dotykem



Obrázek 4.5: Analýza bodu mimo polygony



Obrázek 4.6: Hláška neúspěšného importu

Kapitola 5

Dokumentace tříd a metod

5.1 Algorithms

- *int getPositionWinding(QPointF q, QPolygonF pol)*
  Funkce, která určuje polohu bodu vůči polygonu metodou Winding Number Algorithm. Na vstupu funkce je určovaný bod q a polygon P. Výstupem je celé číslo a to 1(bod leží uvnitř polygonu, na hraně nebo na vrcholu), 0(bod leží vně polygonu) nebo -1(jiné).

- *int getPositionRay(QPointF q, QPolygonF pol)*
  Funkce, která určuje polohu bodu vůči polygonu metodou Ray Crossing Algorithm. Na vstupu funkce je určovaný bod q a polygon P. Výstupem je celé číslo a to 1(bod leží uvnitř polygonu, na hraně nebo na vrcholu) nebo 0(bod leží vně polygonu).

- *int getPointLinePosition(QPointF &q, QPointF &a, QPointF &pol)*
  Tato funkce určuje pozici bodu vůči linii. Na vstupu funkce je určovaný bod a dva body přímky a a b. Ze dvou bodů přímky můžeme určit determinant, jehož hodnotu porovnáváme se zvolenou minimální hodnotou ϵ. Výstupem funkce je celé čislo, které nabívá hodnot 1(pod leží v levé polorovině), 0(bod leží v pravé polorovině) a -1(bod leží na hraně).

- *double get2LinesAngle(QPointF &p1,QPointF &p2,QPointF &p3, QPointF &p4)*
  Tato funkce počítá úhel mezi dvěma hranami. Na vstupu jsou 4 body, které určují dvě přímky. Výstupem je velikost úhlu ve stupních typu double.

5.2 Draw

- *void paintEvent)*
  Metoda, která vykresluje naimportované polygony a bod q. Dále vykreslí polygon, uvnitř kterého se nachází bod Q.

- *void mousePressEvent)*
  Po kliknutí myši uloží souřadnice bodu Q.

- *void setDrawPoint*
  Slouží k zapnutí vykreslování bodu Q. Po spuštění programu lze vytvářet bod q až po stisknutí tlačítka Draw Point. Po obětovném stisknutí je vytváření bodu q opět vypnuté.

- *void clearCanvas*
  Slouží k vyčištění grafického okna od všech polygonů i bodu Q.

- *bool importPolygons*
  Používá se k načtení textového souboru s body polygonů.

5.3 Widget

- *void on_ pushButton_ 3_ clicked)*
  Vyčistí okno aplikace. V aplikaci se jedná o tlačítko "Clear".

- *void on_ pushButton_ clicked)*
  Slouží k tvorbě bodu q. V aplikaci se jedná o tlačítko "Draw Points".

- *void on_ pushButton_ 2_ clicked*
  Spustí výpočet algoritmu pro určení polohy bodu. V aplikaci se jedná o tlačítko "Analyze".

- *void on_ importPolygons_ clicked*
  Při stisknutí tlačítka se otevře dialogové okno pro vyhledání souboru se souřadnicemy bodů polygonů. V aplikaci se jedná o tlačítko "Import Polygon".

Kapitola 6

Závěr

Do vytvořené aplikace lze načíst data polygonů, vytvořit bod Q a spočítat zda bod leží uvnitř polygonu. Aplikace vybarví Polygon, který obsahuje bod Q. V programu jsou vyřešeny i případy, kdy se bod nalézá na hraně nebo vrcholu jednoho nebo více polygonů.

6.1 Neřešené problémy a náměty

- Z časové tísně nebyla v úloze vyřešena tvorba nekonvexních polygonů.
- Data musí být předem upravena, jelikož program má předdefinovanou velikost vykreslovacího okna. V ideálním případě by bylo dobré, kdyby se načtená data sama přízpůsobila velikosti okna.
- Praktické by též bylo rozšířit aplikaci o funkci, jež převádí kartézské souřadnice do souřadnic obrazových.

Literatura

[1] T. Bayer. Geometrické vyhledávání bodu, cvičení č. 1 předmětu Algoritmy v digitální kartografii, 2016. `https://web.natur.cuni.cz`.

[2] T. Bayer. Geometrické vyhledávání bodu, přednáška č. 2 předmětu Algoritmy v digitální kartografii, 2016. `https://web.natur.cuni.cz`.

Příloha A

Zdrojový kód aplikace

Zdrojový kód aplikace je dostupný z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře:*ADK_poskocil_faber*
Podsložka: *u1_pol*

https://github.com/petrposkocil/ADK_poskocil_faber/U1/u1_pol

Obsažené soubory:
1_pol.pro //Qt creator project file
algorithms.h //Header file
draw.h //Header file
widgets.h //Header file
algorithms.cpp //Source code file
draw.cpp //Source code file
widgets.cpp //Source code file
widget.ui //User interface file

Příloha B

Vstupní a výstupní data

Vstupní data jsou dostupná z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře: *ADK_ poskocil_ faber*
Soubor: *imput_ data.txt*

https://github.com/petrposkocil/ADK_poskocil_faber/U1/imput_data.txt

Výstupní data jsou ve fromě grafického znázornění výsledků algoritmů. Polygon do kterého padnou souřadnice bodu Q je vyzobrazen barevnou šrafou.

Příloha C

Aplikace - binární soubor

Aplikace je dostupná z veřejného profilu *petrposkocil* na github.com
Jméno repozitáře: *ADK_ poskocil_ faber*
Soubor: *u1_ pol.exe*

https://github.com/petrposkocil/ADK_poskocil_faber/U1/u1_pol.exe